Automated Geometry Theorem Proving for Human-Readable Proofs

Ke Wang Zhendong Su
Department of Computer Science
University of California, Davis
{kbwang, su}@ucdavis.edu

Abstract

Geometry reasoning and proof form a major and challenging component in the K-12[1] mathematics curriculum. Although several computerized systems exist that help students learn and practice general geometry concepts, they do not target geometry proof problems, which are more advanced and difficult. Powerful geometry theorem provers also exist, however they typically employ advanced algebraic methods and generate complex, difficult to understand proofs, and thus do not meet general K-12 students' educational needs. This paper tackles these weaknesses of prior systems by introducing a geometry proof system, iGeoTutor, capable of generating *human-readable elementary proofs*, *i.e.* proofs using standard Euclidean axioms. We have gathered 77 problems in total from various sources, including ones unsolvable by other systems and from Math competitions. iGeoTutor solves all but two problems in under two minutes each, and more importantly, demonstrates a much more effective and intelligent proof search than prior systems. We have also conducted a pilot study with 12 high school students, and the results show that iGeoTutor provides a clear benefit in helping students learn geometry proofs. We are in active discussions with Khan Academy and local high schools for possible adoption of iGeoTutor in real learning environments.

Video demo: https://www.youtube.com/watch?v=KL0dUb6hKxU

1 Introduction

Geometry is a key, mandatory subject in the secondary school curriculum. An important part of geometry learning is proof writing [Schoenfeld, 1994; Hanna, 1995] which helps train students' logic and deductive reasoning skills. Because of this, it is also one of the most challenging subjects for students. The standard format of a geometry proof problem consists of a given geometry figure, a set of provided constraints and a goal to prove. Students are asked to write a step-by-step deduction using Euclidean axioms. Figure 1 depicts an example problem. The task becomes more difficult when a proof requires auxiliary constructions — adding lines/arcs to the problem figure to help discover a proof — because finding appropriate constructions can be very challenging as one could add any geometric elements anywhere in the problem figure.[2]

Given: Square $ABCD, AP = PD, \angle PAD = 15^\circ$
Goal: $\triangle PBC$ is equilateral

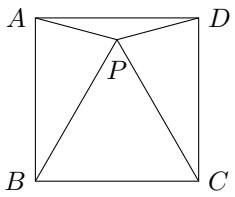

Figure 1: Example geometry proof problem.

Due to the aforementioned reasons, we envision that a powerful automated geometry proof system can benefit students, because it provides a foundation to help them practice their problem-solving skills. It can also facilitate the current nascent paradigm shift in education toward massively online and personalized learning, targeting the K-12 students.

Background and Related Work Although computer-based geometry learning tools have been successfully applied in education, many of these systems focus on simple tasks, such as drawing and calculation, and do not deal with geometry proofs (*e.g. Cabri II Plus*, *Sketchpad*, and *Geometry Expression*). The few exceptions [Gao and Zhu, 1999; Janičić, 2006; Narboux, 2007] are all limited by their underlying geometry theorem proving algorithms. For example, *GeoProof* [Narboux, 2007] and *GEX* [Gao and Zhu, 1999] adopt algebraic methods [Wu, 1986; Kutzler and Stifter, 1986; Buchberger *et al.*, 1988] for geometry theorem proving. Although these methods are powerful, they rely on algebraic theories and can only decide the validity of geometry statements [Jiang and Zhang, 2012]. *GCLC* [Janičić, 2006] is built on top of the area method [Chou *et al.*, 1993], which utilizes a specialized set of *nonstandard axioms* unfamiliar to students

[1]The term "K-12" is commonly used in the United States and Canada to collectively refer to primary and secondary education.

[2]The interested reader may attempt the problem in Figure 1. Trigonometry is not needed.

learning standard Euclidean geometry. For the above reasons, none of the systems is suitable for general K-12 students.

The goal of our work is to help students learn geometry proof reasoning, thus our primary design objective is *proof readability*. By readable proofs, we mean proofs that are formulated using Euclidean geometry axioms taught in schools. The most challenging technical obstacle that we face is auxiliary construction, which is necessary for many problems. Matsuda *et al.* [Matsuda and Vanlehn, 2004] propose the first technique for auxiliary construction. Their idea is to only add elements to satisfy premises of a postulate whose consequence "matches" a goal. Although Matsuda *et al.* have made an impressive first step toward generating human-readable proofs using standard Euclidean axioms, their approach is ineffective and can cause combinatorial explosion when multiple construction steps are needed.

Template-based Geometry Theorem Proving To address the weaknesses of prior systems, we introduce a general *template matching-based approach* to tackle the auxiliary construction challenge for geometry theorem proving. Our key observation is that problem solving typically involves the use and trained familiarity with a number of important problem solving strategies. At the high-level, we (1) distill a set of *strategies* in the form of *template figures* from studying around 20 problems frequently used for competition training, and (2) develop a *template matching scheme* to recover templates from a given figure.

We have realized our approach as a new automated geometry theorem prover, iGeoTutor and extensively evaluated it. Our problem corpus has 77 problems in total, including ones cited in prior work and from Math competitions. Evaluation results show that iGeoTutor is very effective — it solves all but two problems in under two minutes each. Moreover, its construction process is significantly more efficient than GRAMY's on the same problem set (*cf.* Table 2, Section 3.2 for the detailed comparison). We have also performed a test pilot with 12 high school students, and found iGeoTutor is effective in helping student to learn geometry proof. Based on all the evaluation results, we expect iGeoTutor will impact K-12 geometry education by helping students become strong problem solvers and self learners.

Contributions Our main contributions are:

- We propose a novel, template matching-based approach that is powerful and produces human-readable proofs.
- We realize our approach in iGeoTutor, which can effectively solve nearly all problems in our collection using six simple templates.
- We present details of our extensive evaluation of iGeoTutor's performance in solving geometry proof problems and its effectiveness in helping students learn.

The remainder of this paper is organized as follows. Section 2 describes our methodology and approach in designing iGeoTutor, and Section 3 details our evaluation results. We conclude in Section 4 with a discussion of future work .

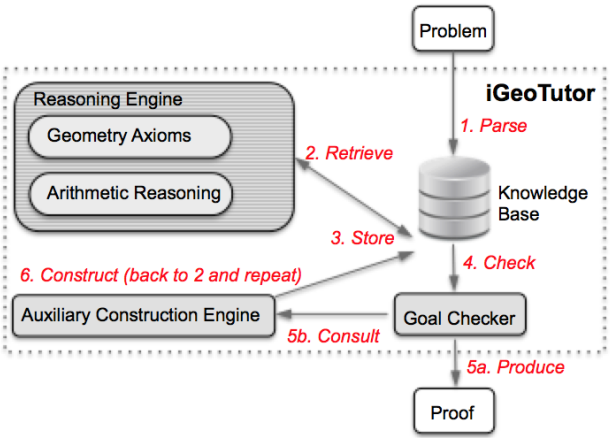

Figure 2: iGeoTutor system architecture.

2 Methodology and Approach

This section presents the design of iGeoTutor.[3] Central to our system is a novel approach for auxiliary construction, that forms the foundation for our work. In particular, we introduce the concept of templates and template-matching scheme.

2.1 System Architecture

Figure 2 shows the overall system architecture of iGeoTutor. We discuss the other key components in this section, and defer the discussion of the core component — the "Auxiliary Construction Engine" — to later sections. We next give a step-by-step description of iGeoTutor's workflow:

(1) **Parsing**: iGeoTutor parses a given geometry figure from its pictorial representation to an internal representation using first-order predicates and store them in the Knowledge Base. This paper omits the details of this step as it is not the focus of this work.

(2) **Retrieving**: The Reasoning Engine retrieves the facts from the Knowledge Base and derives new facts using Geometry Axioms and Arithmetic Reasoning rules.

(3) **Storing**: iGeoTutor stores any derived facts to the Knowledge Base.

(4) **Checking**: iGeoTutor checks whether the goal is in the Knowledge Base.

(5) **Result**: Two cases:

  (a) **Producing**: If the goal is in the Knowledge Base, iGeoTutor terminates and produces a proof.

  (b) **Consulting**: Otherwise, it consults Auxiliary Construction Engine for additional facts according to the existing geometric configuration in the problem figure.

(6) **Constructing**: iGeoTutor stores the facts about an auxiliary construction in the Knowledge Base and diverts its execution to step (2) and repeat.

[3] In our current prototype, we focus on problems involving line segments and angles, and do not consider those involving circles, arcs, and solid geometry, which we leave for future work.

Knowledge Base Given a geometry diagram and a set of associated constraints, iGeoTutor uses first-order predicates for its internal representation. The predicates can be classified into three categories[4]:

- *Geometrical elements*, such as line(A,B) and angle(A,B,C);
- *Position relation*, such as parallel(line(A,B),line(C,D));
- *Quantity relation*, such as lineEqual(line(A,B),line(C,D)).

For example, the problem in Figure 1 is represented internally as follows:

- Given: square(A,B,C,D), line(A,P), line(B,P), line(C,P), line(D,P), lineEqual(line(A,P),line(D,P)) and angleEqual(angle(P,A,D),$15°$).
- Prove: equilateralTriangle(B,P,C).

Apart from the problem representation, the Knowledge Base also stores *derived* facts.

Geometry Axioms Geometry Axioms refer to definitions, postulates, and previously proved theorems within the scope of Euclidean geometry. Each axiom consists of premises and a consequence, which are represented internally using the predicates introduced earlier. For example, the "Side-Angle-Side" postulate can be expressed as the following Horn clause:

triangleCongruent(triangle(A,B,C),triangle(D,E,F)) ←
triangle(A,B,C) ∧ triangle(D,E,F) ∧
lineEqual(line(A,B),line(D,E)) ∧ lineEqual(line(B,C),line(E,F)) ∧
angleEqual(angle(A,B,C),angle(D,E,F)).

Arithmetic Reasoning This component refers to the arithmetic reasoning capability incorporated into iGeoTutor. In particular, we employ an off-the-shelf state-of-the-art *SMT solver Z3* [De Moura and Bjørner, 2011; Microsoft Research,] to resolve the arithmetic constraints when needed. We interface iGeoTutor and Z3 in three phases: (1) iGeoTutor converts all its quantitative predicates in the knowledge base to Z3 commands in Z3's script language; (2) the Z3 checker executes the Z3 script to infer additional quantitative constraints; and (3) we convert Z3 inferred constraints back to iGeoTutor's internal predicates to store in the Knowledge Base.

Reasoning Engine The Reasoning Engine applies the Geometry Axioms and the Arithmetic Reasoning rules on facts stored in the Knowledge Base to deduce additional facts. iGeoTutor employs forward chaining [Chou *et al.*, 2000] to update the given constraints. For example, on the internal representation of the problem in Figure 1, the Reasoning Engine applies the aforementioned Horn clause of the "Side-Angle-Side" postulate to derive, among others, the following:

- triangleCongruent(triangle(A,B,P),triangle(D,C,P))
- lineEqual(line(B,P),line(C,P))

Clearly, the goal of the problem cannot be reached given the original problem figure; auxiliary constructions are needed. We explain next how we tackle this difficulty.

[4]The interested reader may refer to http://www.cs.ucdavis.edu/~su/igeotutor.html for the complete list of predicates and axioms.

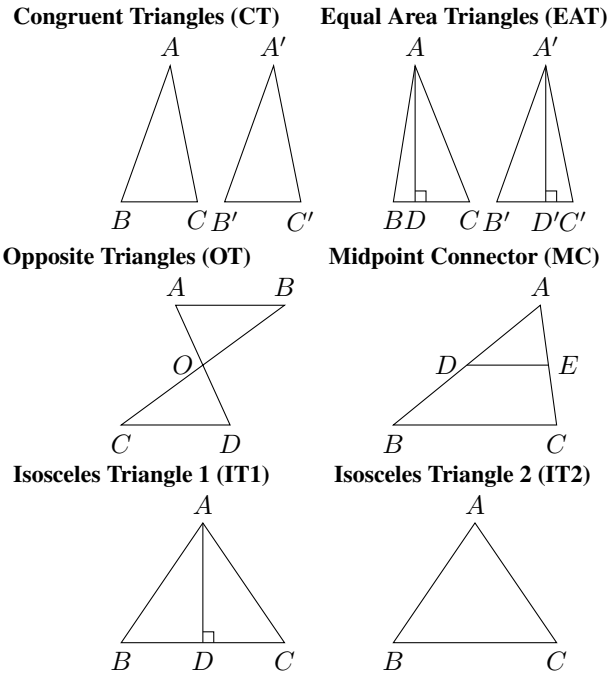

Figure 3: The six distilled template figures.

2.2 Auxiliary Construction

This section describes our template-based technique for auxiliary construction. We introduce the six templates that we use, illustrate via examples how templates guide auxiliary construction, and present our detailed construction procedure.

2.2.1 Templates

As mentioned earlier, templates intuitively capture problem solving strategies. Templates can be adapted to solve particular classes of geometry problems. We represent each template by a common geometry shape that is used to illustrate one standard Euclidean axiom. Take the "IT2" template in Figure 3 for example, this template captures the strategy of exploiting the equivalence of a pair of angles or segments. A key challenge is how to discover these templates, which we discuss next.

We distill a small set of template figures from our close examination of around 20 training problems. We discover that over all these problem instances, any auxiliary construction that leads to an eventual proof is used mostly to *realize some specific geometry shape*. Perhaps quite surprisingly, only six shapes are used frequently. Figure 3 depicts these six template figures, which are quite basic. They all concern line or angle congruence, which in retrospect may not be surprising as the concept of congruence underlies most geometry proof problems. This study confirms our hypothesis that a few common, effective strategies exist for geometry proof problems. Next, we show how to use these templates to steer proof search.

2.2.2 Template-Matching Example

We use the example problem in Figure 1 to illustrate how to use the template figures to discover auxiliary constructions.

As mentioned in Section 2.1, the given problem figure suf-

Statement	Reason
$\triangle AQB \cong \triangle APD$	SAS
$\angle BAQ = 15^\circ, \angle BQA = 150^\circ$	CPCTC[5]
$\angle QAP = 60^\circ$	Subtraction
$\angle AQP = \angle APQ = 60^\circ$	Isosceles Triangle
$\angle BQP = 150^\circ$	Subtraction
$\triangle AQB \cong \triangle PQB$	SAS
$AB = BP$	CPCTC
$BC = BP = CP$	Transitive Equality

Table 1: Sample proof.

- Template: Congruent Triangles
- Matching Instance: AB, $\triangle APD$, $AB = AD$
- Auxiliary Construction: Find a point Q such that $\triangle AQB$ is congruent to $\triangle APD$
- Realization: Introduce a point Q, with $\angle BAQ = \angle PAD$ and $AQ = AP$, and connect BQ

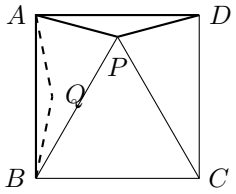

Figure 4: Applying the "CT" template (step 1).

fices to help derive $BP = PC$, but to prove $BP = BC$ or $PC = BC$, we must add auxiliary constructions. First, assume that we decide to apply the "CT" template on the existing $\triangle APD$ and AB. To complete the "CT" template, we construct a new point Q to make $\triangle AQB$ congruent to $\triangle APD$ as shown in Figure 4. Second, assume that we decide to use the "IT2" template to complete an isosceles $\triangle AQP$ by connecting P and Q as depicted in Figure 5. With these constructions, we can discover a proof via forward-chaining.

Table 1 shows a short version of the proof iGeoTutor discovers by applying the "CT" template and the "IT2" template. There is an obvious alternative algebraic solution by utilizing $\tan 15^\circ$ and the *Pythagorean Theorem* to calculate the length of BP. However, the sample proof discovered by iGeoTutor uses only standard Euclidean axioms (not any trigonometry), which is more elegant and suitable for students learning elementary geometry. Note that the problem has a number of alternative solutions; here we illustrate one of them.

Next, we focus on two important questions: (1) How to decide which template to apply? and (2) How to select the existing geometric elements to apply for the template? To this end, we introduce our template matching scheme next.

2.2.3 Template-Matching Scheme

We discuss our high-level template matching process first and then introduce key optimizations to make it practical.

[5]"Corresponding Parts of Congruent Triangle are Congruent"

- Template: Isosceles Triangle (2)
- Matching Instance: AP, AQ, $AQ = AP$
- Construction: Complete the triangle $\triangle APQ$
- Realization: Connect P and Q

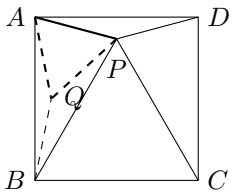

Figure 5: Applying the "IT2" template (step 2).

High-Level Template Matching Process Let each template t be a pair $\langle G, C\rangle$, where G is the template figure and C the set of constraints satisfied by G. Let $\wp(C)$ denote the power set of C. For each subset $S \in \wp(C)$, construct the tuple $T_s = \langle G_s, C \setminus S\rangle$, where G_s is a sub-figure of G induced by S and $\setminus$ is set difference.

At the high-level, a template matching step traverses $S \in \wp(C)$ and unifies the constraints in each subset S against the constraints C_p from the current problem configuration, *i.e.* the constraints C_p and figure G_p. If a unification check succeeds, we *overlay* G_s on G_p to add any missing elements G_m from G and constraints $C \setminus S$; the pair $\langle G_m, C \setminus S\rangle$ constitute an auxiliary construction.

Key Optimizations We introduce two important heuristics to optimize the high-level search procedure. One prioritizes the search of the subsets, while the other opportunistically leverages the goal and any of its derived constraints for matching.

Optimization 1: Search Prioritization A template may induce a large number of subsets to unify against, thus it is important to order them such that the more "profitable" subsets are checked earlier. To this end, we develop two techniques.

First, for each template $\langle G, C\rangle$, we call a set $S \in \wp(C)$ a *minimum sufficient set* iff

$$S \models C \;\; \wedge \;\; \forall S' \subset S \;\; S' \not\models C$$

i.e., S is a minimal, logically equivalent subset of C. The intuition is that a minimum sufficient set induces a *complete* figure *w.r.t.* the template and should be matched first. For example, the constraints for the problem figure in Figure 5 unify with the minimum sufficient set (*w.r.t.* the "IT2" template) $\{AQ = AP\}$, specifying that two equivalent lines share a common end-point. In contrast, the matching example in Figure 4 is achieved via a non-minimum sufficient set. For matching purposes, minimum sufficient sets are given higher priorities than the other subsets.

Second, we introduce a ranking heuristic to target the non-minimum sufficient sets. The basic intuition behind the heuristic is to maximize "gain" of a subset — *few added elements* lead to *many derived constraints*. Thus, we define "gain" as the ratio of the number of derived constraints (from constraints induced by the added elements and the given constraints in the problem figure) and the number of added elements to realize

Algorithm 1: Construction search via template matching

```
1  procedure SearchConstruction(int depth, Knowledge knowledge)
2      begin
3          if depth < maxDepth then
4              matchedTemplates
               ← TemplateSearchProcedure(knowledge)
5              foreach template in matchedTemplates do
6                  construction ← SynthesisConstruction(template)
7                  knowledge.addConstruction
                   (construction)
8                  if knowledge.reasoning() then
9                      return knowledge.getProof() else
10                         return SearchConstruction
                           (depth + 1, knowledge)
```

Algorithm 2: Template matching procedure

```
1  function TemplateSearchProcedure(Knowledge knowledge)
2      begin
3          goalDerived
           ← FindingTruePropositions(knowledge.getGoal(),
           knowledge.getConditions())
4          backwardTemplates
           ← BackwardMatching(knowledge.getGoal(), goalDerived,
           knowledge.getConditions())
5          forwardTemplates
           ← ForwardMatching(knowledge.getConditions())
6          matchedTemplates
           ← MixAndRank(backwardTemplates, forwardTemplates)
7          return matchedTemplates
```

a template figure. Our heuristic is to rank the non-minimum sufficient sets *w.r.t.* their "gain". Later, we show empirically that the heuristic is effective and performs better than the pure random ranking scheme.

Optimization 2: Goal-Directed Matching To complement the first optimization, we introduce another novel technique to further improve the accuracy and effectiveness of the template matching procedure. In particular, before template matching, the reasoning engine incorporates the problem goal into its knowledge base and performs exhaustive forward chaining until it derives no new facts (*i.e.* reaching the *constraint closure*). For template matching, all constraints in the closure are considered. Our insight is to exploit every fact associated with the given geometric figure. Note that this does not invalidate the soundness of our system because the goal and any goal-related facts are removed from the knowledge base in the actual reasoning phase — the sole purpose of this optimization is to aid template matching for discovering auxiliary constructions.

Putting Everything Together As shown in Algorithm 1, our construction search procedure is cast as *depth-first search*. It begins by attempting to find all matching templates through the function *TemplateSearchProcedure w.r.t.* the current problem configuration (line 4). While traversing each matched template, it synthesizes the respective auxiliary constructions and incorporates them to the original problem figure represented by knowledge (lines 5-7). Next, the reasoning engine exhaustively derives new facts (line 8). If it finds the goal, the construction procedure exits with the discovered proof (line 9). Otherwise, it invokes a recursive call to continue the search (line 10). This procedure repeats until a proof is found.

Note that in the function *TemplateSearchProcedure* shown in Algorithm 2, *FindingTruePropositions* helps infer all facts after the goal has been incorporated (line 3). Subsequently *BackwardMatching* uses the goal and any goal-derived facts (along with the existing facts) to find and match templates (line 4). Finally it ranks and returns all matching templates (lines 6-7).

3 Evaluation

First, we present our empirical evaluation of iGeoTutor's effectiveness. It also includes a comparison with GRAMY, the previously state-of-the-art system. Second, we describe a field study for assessing iGeoTutor's effectiveness in helping students learn geometry proofs.

3.1 Test Corpus

Our test corpus contains 77 problems in total (please refer to http://www.cs.ucdavis.edu/~su/igeotutor.html for the full list of problems and their descriptions). The problems have been gathered from various sources, including 22 from the work on GRAMY [Matsuda and Vanlehn, 2004][6] for comparison. The rest of the problems are from an archive of classical geometry proof problems online [Liu, 2011] and two Chinese textbooks [Shen, 2006; Zhou, 2004], both of which are popular practice materials for the Chinese Mathematics Olympiad. A number of the problems involve arithmetic calculations. As mentioned earlier, iGeoTutor supports arithmetic, but not trigonometry, and restricts its search for auxiliary constructions to rely only on the supplied constraints when given a *measurement problem*, *i.e.* a problem that asks for the size of an angle or the length of a segment.

3.2 Evaluation Setup and Results

iGeoTutor runs on a workstation with a third generation Intel Core i7-3770 processor and 16GB RAM. We present two sets of results: (1) general results on all 77 problems, and (2) detailed results on the 22 problems from GRAMY.

3.2.1 iGeoTutor Performance

First, out of the 77 problems, iGeoTutor solves 75 in under two minutes each (we defer to Section 3.4 to discuss the two problems that iGeoTutor fails to solve). All constructions are accomplished using the six templates from Section 2. For each of the successfully solved problems, iGeoTutor matches *three or fewer* templates before discovering a proof. Moreover, 17 problems are solved *only* by incorporating their goals, demonstrating the importance of our goal-directed search heuristic.

Second, Table 2 presents detailed results for both iGeoTutor and GRAMY on the 22 problems from GRAMY. We discuss the results for iGeoTutor first. In the table,

- *Depth* refers to the depth of iGeoTutor's construction search procedure (Algorithm 1). In parentheses, we also specify each construction's matching type: (1) *m* for matching via a minimum sufficient set, and (2) *n* for matching via a non-minimum sufficient set. For example,

[6] The original list contains 32 problems, but nine are from textbooks that we do not have access to, and one requires trigonometry.

Problem	iGeoTutor					GRAMY		
	Depth	Length	States	Time(s)	Gain	Depth	Length	States
$P001$	2 (n,m)	0	2	1	26	6	6	130
$P002$	2 (n,m)	0	6	7	5.1	✗	✗	✗
$P003$	2 (n,m)	0	2	2	11.5	✗	✗	✗
$P004$	2 (n,m)	0	31	87	32.9	✗	✗	✗
$P005$	1 (m)	0	1	1	1	4	3	4
$P006$	1 (m)	0	1	1	1	5	4	15
$P007$	1 (n)	0	2	3	.4	5	5	198
$P008$	1 (m)	1	4	2	1	5	5	313
$P009$	1 (n)	1	4	3	1.6	6	9	48
$P010$	1 (n)	0	4	2	3.1	6	8	23
$P011$	2 (n,m)	0	2	2	3.2	6	14	112
$P012$	1 (m)	3	3	1	1	–	30	78
$P013$	1 (n)	0	1	1	3.5	7	18	26
$P014$	1 (m)	0	2	2	1	8	7	80
$P015$	1 (n)	0	3	14	68	9	10	13
$P016$	1 (m)	0	1	2	1	9	19	85
$P017$	2 (m,m)	0	4	8	1	–	4	36
$P018$	1 (m)	0	1	1	1	–	6	9
$P019$	1 (n)	1	1	8	2.3	–	7	9
$P020$	1 (n)	0	1	2	6	–	13	52
$P021$	1 (n)	0	23	34	3.1	–	–	–
$P022$	3 (n,n,m)	0	47	106	1.8	✗	✗	✗

Table 2: Evaluation results.

"3(n,n,m)" for $P022$ indicates that it requires 3 constructions, with the first two being "n" and the last "m".

- *Length* refers to the number of axiom applications until the first successful templating matching attempt (*i.e.* all premises of a template matching are satisfied).
- *States* refers to the total number of states expanded before reaching a proof. In general, for geometry theorem proving, a state consists of a problem figure, all true propositions associated with the figure's geometrical elements, and goals to prove. For iGeoTutor, the traversered states correspond to the different instantiations of template matching for auxiliary construction.
- *Time* refers to the total time in seconds that iGeoTutor takes to produce a proof.
- *Gain* refers to the performance gain of our search prioritization heuristic over when it is disabled (*i.e.* using purely random search).

From the table, it is clear that iGeoTutor performs well and the search prioritization heuristic is effective.

3.2.2 iGeoTutor vs. GRAMY

Since GRAMY shares the same high-level goal as iGeoTutor and is the previously state-of-the-art system, we evaluate how the two systems compare. Table 2 also includes GRAMY's performance results on the identical problem set. The performance numbers for GRAMY are extracted from its publication [Matsuda and Vanlehn, 2004], where '–' indicates unavailable data.

Because the GRAMY system is unavailable and it was evaluated on older hardware, we do not include any of its timing data, but rather focus on the other data that reflect the complexity of its proof search for a fair comparison. Thus, we focus on the common, similar metrics as in iGeoTutor: "Depth", "Length", and "States". In particular, (1) *Depth* refers to the total number of invocations of GRAMY's construction procedure before it finds a proof as its proof procedure is formalized as a breadth-first search [Matsuda and Vanlehn, 2004], (2) *Length* refers to the number of axiom applications needed to realize the first successful construction, and (3) *States* refers to the total number of expanded states before finding a proof. Because GRAMY also employs backward chaining, so state changes are not only due to attempted constructions but also changes in the problem goal (by adding sub-goals).

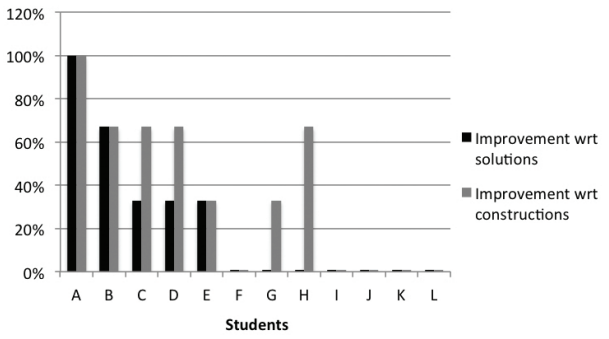

Figure 6: Pilot study results.

iGeoTutor successfully solved all four problems that GRAMY failed to solve (marked by '✗') — two ($P002$ and $P003$) took little time and had relatively simple proof search, while the other two ($P004$ and $P022$) took relatively longer and had more complex proof search. Table 2 also shows that iGeoTutor's proof search is significantly more effective. For example, in terms of the number of expanded states, the most direct measure on search complexity, iGeoTutor exhibits orders of magnitude improvement over GRAMY.

3.3 Pilot Study

Participants For the study, we have recruited 12 students from the local high school. All students are in ninth grade, the typical grade level for the Geometry standards in California.

Procedure First, the participants took a pre-test. Then, they used iGeoTutor to explore those problems they failed to solve on the pre-test. Finally they took a post-test. Both the pre-test and post-test contain three geometry proof problems (please refer to http://www.cs.ucdavis.edu/~su/igeotutor.html for the two tests). Each participant was given 30 minutes to complete each of the three parts of the study.

Results Figure 6 summarizes the results of our study. Specificially, it shows each student's performance improvement on the post-test over the pre-test: dark bars illustrate improvement measured in terms of successfully solved problems, while gray bars illustrate improvement measured in terms of correctly identified constructions. We have run a paired t test to compare each participant's performance (in terms of number of solved problems) across the two test sets. The results show that students' post-test performance is significantly better than that of the pre-test [$t(10) = 2.3834, p = 0.0384$], thus indicating that iGeoTutor clearly helps students learn.

3.4 iGeoTutor's (In)Completeness

As aforementioned, iGeoTutor failed to solve two ($P073$ and $P076$) of the 77 problems (please refer to http://www.cs.ucdavis.edu/~su/igeotutor.html for details on these two problems). This section discusses the reasons for iGeoTutor's proof search to be incomplete and possible solutions. We identify two sources of incompleteness:

Missing Axioms For $P076$, iGeoTutor found two constructions that could successfully lead to the solution ($\angle BAC = 40°$). iGeoTutor failed even with the right auxiliary constructions because it was unable to exploit a property of four points on a circle to derive $\angle ACE = \angle ADE$. As mentioned earlier, iGeoTutor is only given the axioms for dealing with non-circle problems and therefore it fails to solve the problem that requires circle-related axioms. A simple solution is to equip iGeoTutor with all elementary geometry axioms.

Missing Templates The six templates that we distilled from around 20 problems are sufficient to help solve nearly all problems in our corpus. We do not, however, claim that iGeoTutor's construction search procedure is complete. $P073$ is the only problem from the corpus that iGeoTutor fails to solve due to incomplete templates — one step of the construction is not covered by the six templates. However, the figure was very similar to "OT" and could be considered its variant, which may be obtained by relaxing the midpoint constraint for "OT". For cases like this, one possible direction is to support a set of shape variants for each template while fixing some original constraints. This would allow more flexible and effective problem solving strategies.

4 Conclusion

This paper has presented a novel, practical template-based construction search algorithm for automated geometry theorem proving. Our evaluation demonstrates its power in geometry problem solving and its effectiveness in helping students to learn geometry proof. For future work, we plan to explore three main directions. First, we plan to utilize statistical predictions trained on our problem corpus to suggest good candidates for template matching. Second, we would like to extend our work to support problems that involve circles and arcs, and in solid geometry. Third, we plan to further evaluate and refine our system in real learning environments, and improve students' learning experience.

References

[Buchberger *et al.*, 1988] B. Buchberger, G. Collins, and B. Kutzler. Algebraic methods for geometric reasoning. *Annual Reviews in Computer Science*, 3:85–120, 1988.

[Chou *et al.*, 1993] Chou, Gao, and Zhang. Automated production of traditional proofs for constructive geometry theorems. In *LICS: IEEE Symposium on Logic in Computer Science*, 1993.

[Chou *et al.*, 2000] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. A deductive database approach to automated geometry theorem proving and discovering. *J. Autom. Reasoning*, 25(3):219–246, 2000.

[De Moura and Bjørner, 2011] Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: Introduction and applications. *Communications of the ACM*, 54(9):69–77, 2011.

[Gao and Zhu, 1999] XS Gao and C Zhu. Building dynamic mathematical models with Geometry Expert, III, A geometry deductive database. In *Proceedings of Asian Technology Conference in Mathematics*, pages 153–162, 1999.

[Hanna, 1995] Gila Hanna. Challenges to the importance of proof. *For the Learning of mathematics*, pages 42–49, 1995.

[Janičić, 2006] Predrag Janičić. GCLC — A tool for constructive Euclidean geometry and more than that. In *Mathematical Software-ICMS 2006*, pages 58–73. Springer, 2006.

[Jiang and Zhang, 2012] Jianguo Jiang and Jingzhong Zhang. A review and prospect of readable machine proofs for geometry theorems. *Journal of Systems Science and Complexity*, 25(4):802–820, 2012.

[Kutzler and Stifter, 1986] B. Kutzler and S. Stifter. On the application of Buchberger's algorithm to automated geometry theorem proving. *JSC*, 2(4):389–397, December 1986.

[Liu, 2011] Wenchuanm Liu. Classical elementary geometry problems (in chinese), 2011.

[Matsuda and Vanlehn, 2004] Matsuda and Vanlehn. GRAMY: A geometry theorem prover capable of construction. *JAR: Journal of Automated Reasoning*, 32, 2004.

[Microsoft Research,] Microsoft Research. Getting started with Z3: A guide. URL: http://rise4fun.com/Z3/tutorial/guide.

[Narboux, 2007] Julien Narboux. A graphical user interface for formal proofs in geometry. *Journal of Automated Reasoning*, 39(2):161–180, 2007.

[Schoenfeld, 1994] Alan H Schoenfeld. What do we know about mathematics curricula? *The Journal of Mathematical Behavior*, 13(1):55–80, 1994.

[Shen, 2006] Wenxuan Shen. *Compendium Proof Methods for Elementary Geometry (in Chinese)*. Harbin Institute of Technology Press, Harbin, 2 edition, 2006.

[Wu, 1986] W.-T. Wu. Basic principles of mechanical theorem proving in geometry. *Journal of Automated Reasoning*, 2:221–252, 1986.

[Zhou, 2004] Chunli Zhou. *Elementary Geometry in Middle School Math Competition (in Chinese)*. China Supplies Press, Beijing, 1 edition, 2004.